Scientific Excellence • Resource Protection & Conservation • Benefits for Canadians
Excellence scientifique • Protection et conservation des ressources • Bénéfices aux Canadiens

AN ALGORITHM FOR BOTTOM WINDOW ECHO PROCESSING

Ross Shotton

Program Coordination and Economics Branch
Department of Fisheries and Oceans
Box 550
Halifax, Nova Scotia
B3J 2S7

April 1991

CANADIAN MANUSCRIPT REPORT OF FISHERIES AND AQUATIC SCIENCES

No. 2113

Fisheries and Oceans Pêches et Océans

Canada

91-056841

Canadian Manuscript Report of Fisheries and Aquatic Sciences

Manuscript reports contain scientific and technical information that contributes to existing knowledge but which deals with national or regional problems. Distribution is restricted to institutions or individuals located in particular regions of Canada. However, no restriction is placed on subject matter, and the series reflects the broad interests and policies of the Department of Fisheries and Oceans, namely, fisheries and aquatic sciences.

Manuscript reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report is abstracted in *Aquatic Sciences and Fisheries Abstracts* and indexed in the Department's annual index to scientific and technical publications.

Numbers 1–900 in this series were issued as Manuscript Reports (Biological Series) of the Biological Board of Canada, and subsequent to 1937 when the name of the Board was changed by Act of Parliament, as Manuscript Reports (Biological Series) of the Fisheries Research Board of Canada. Numbers 901–1425 were issued as Manuscript Reports of the Fisheries Research Board of Canada. Numbers 1426–1550 were issued as Department of Fisheries and the Environment, Fisheries and Marine Service Manuscript Reports. The current series name was changed with report number 1551.

Manuscript reports are produced regionally but are numbered nationally. Requests for individual reports will be filled by the issuing establishment listed on the front cover and title page. Out-of-stock reports will be supplied for a fee by commercial agents.

Rapport manuscrit canadien des sciences halieutiques et aquatiques

Les rapports manuscrits contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles, mais qui traitent de problèmes nationaux ou régionaux. La distribution en est limitée aux organismes et aux personnes de régions particulières du Canada. Il n'y a aucune restriction quant au sujet; de fait, la série reflète la vaste gamme des intérêts et des politiques du ministère des Pêches et des Océans, c'est-à-dire les sciences halieutiques et aquatiques.

Les rapports manuscrits peuvent être cités comme des publications complètes. Le titre exact paraît au-dessus du résumé de chaque rapport. Les rapports manuscrits sont résumés dans la revue *Résumés des sciences aquatiques et halieutiques*, et ils sont classés dans l'index annuel des publications scientifiques et techniques du Ministère.

Les numéros 1 à 900 de cette série ont été publiés à titre de manuscrits (série biologique) de l'Office de biologie du Canada, et après le changement de la désignation de cet organisme par décret du Parlement, en 1937, ont été classés comme manuscrits (série biologique) de l'Office des recherches sur les pêcheries du Canada. Les numéros 901 à 1425 ont été publiés à titre de rapports manuscrits de l'Office des recherches sur les pêcheries du Canada. Les numéros 1426 à 1550 sont parus à titre de rapports manuscrits du Service des pêches et de la mer, ministère des Pêches et de l'Environnement. Le nom actuel de la série a été établi lors de la parution du numéro 1551.

Les rapports manuscrits sont produits à l'échelon régional, mais numérotés à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page du titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Canadian Manuscript Report of
Fisheries and Aquatic Sciences No. 2113

April 1991

AN ALGORITHM FOR BOTTOM WINDOW ECHO PROCESSING

Ross Shotton

Program Coordination and Economics Branch
Department of Fisheries and Oceans
Box 550
Halifax, Nova Scotia
B3J 2S7

(c) Ministry of Supply and Services Canada 1991
Cat. No. Fs 97-4/2113E ISSN 0706-6473

Correct citation for this publication is:

Shotton, Ross. 1991. An algorithm for bottom window echo processing. Can. MS. Rep. Fish. Aquat. Sci. No. 2113. 21pp.

TABLE OF CONTENTS

ABSTRACT	iii
RESUME	iii
PREFACE	iii
1. INTRODUCTION	1
2. INPUT DATA	2
3. PROCESSING PROCEDURES	2
3.1 INTRODUCTION	2
3.2 *FALSE BOTTOMS*	3
3.3 *LOST BOTTOMS*	4
4. MONITORING OF PROCESSING	4
5. DISCUSSION	5
6. LITERATURE CITED	5

Appendices

I Program Variables	6
II Program Logic Description	6

ABSTRACT

Shotton, Ross. 1991. An algorithm for bottom window echo processing. Can. MS. Rep. Fish. Aquat. Sci. No. 2113: 21pp.

Acoustic data that are used to assess fish stock abundance must be filtered to ensure that echoes from the seafloor are not attributed to fish, hence causing errors in fish abundance estimates. This error, termed "*lost bottom*," is most likely when the bottom-echo intensity rises slowly and does not reach the threshold value, or reaches it only after an extended period. This usually occurs over steeply sloping seabeds. Similarly, situations in which the density of fish in the reverberation volume is sufficient to trigger a "*false bottom*" will lead to an underestimate of fish abundance as echoes from the fish school will be rejected as bottom echoes.

A processing algorithm is described which attempts to detect the occurrence of *lost* and *false bottoms*. The algorithm allows the operator to carefully consider the consequences of "Bottom Window" correction to ensure that the program parameters remain valid for the situation that is being corrected. Experience shows that no fail-safe methods has yet been developed. This underlines the need for careful post-cruise analysis of fish echo data prior to estimating total acoustic back-scattering cross sections for survey areas.

Keywords: Bottom Window echo processing, acoustic fish stock assessment, echo processing, false bottom, lost bottom.

RESUME

Shotton, R., 1991. An algorithm for bottom window echo processing. Can. MS. Rep Fish. Aquat. Sci. No. 2113: 21pp.

Les données acoustiques qui servent à évaluer des stocks de poissons doivent être filtrées afin d'assurer que les échos du fond ne soient pas attribués au poissons, causant ainsi des erreurs dans l'estimation de l'abondance des poissons. Ce genre d'erreur, appelé "fond perdu," se produit surtout lorsque l'intensité de l'écho du fond accroît lentement et n'atteint pas la valeur-seuil, ou atteint ce seuil seulement après une période prolongée. Ceci se produit d'habitude sur les fonds escarpés. De la même manière, les situations dans lesquelles la densité de poissons dans le volume de réverbération est suffisament elevée pour causer un "faux fond" mèneront à une sous-estimation de l'abondance parce que les échos du banc de poissons seront rejetés comme étant les échos provenant du fond.

Un algorithme est décrit qui tente à déceler les faux fonds et les fonds perdus. L'algorithme permet à l'opérateur de considérer les conséquences des corrections "Fenêtre de fond" pour s'assurer que les paramètres du programme soient valides pour la situation. L'expérience démontre qu'aucune méthode dévelopée jusqu'à date écarte toute possibilité d'erreur. Ceci souligne la nécessité d'une analyse soigneuse des données d'échos après la croisière et avant l'estimation des coéfficients d'échos pour les zones des relevés.

PREFACE

This report has been prepared at the request of users of the ECOLOG fisheries acoustic system in the Gulf Region of the Department of Fisheries and Oceans. Its purpose is to document the logic and procedures used in one of a series of data processing algorithms written to analyse data recorded by the ECOLOG acoustic system. I have attempted to use this opportunity not only to satisfy the need for a record of how data collected from past acoustic survey cruises have been processed, but also to discuss the theory and problems associated with accurate bottom detection, or "Bottom Window" echo processing.

The software described herein was from time to time modified with ad hoc code to handle unique or rare acoustic situations encountered in processing echo data. While it was my preference that discussion of such code be excluded from the report, it has been included at the request of DFO, Gulf Region, to ensure that a faithful account be recorded of past data processing methods.

Input data to this program consisted solely of echo intensity measurements taken from pulses sonifying fish or fish schools. These data were separated from the full echo record by a different program used prior to the "Bottom Window" algorithm. Acoustic data from the "Bottom Window" algorithm program became the input data for the program that determined the total acoustic backscattering cross section in a particular survey area.

1. INTRODUCTION

In digital acoustic systems the elapsed time between signal generation and echo return is used to calculate bottom depth, using the speed of sound in seawater. An echo is usually attributed to the sea bed when its intensity exceeds a certain level, which is usually empirically set. When an echo is judged to be from the seafloor, no further processing of the echo record from that particular transmission usually occurs. Among the many phenomena that are the bane of fisheries acoustics are two arising from the manner in which an acoustic system determines if an echo intensity measurement is from the sea floor.

It is in the empiricism in deciding that an echo is from the seafloor that problems can occur. If the number of fish that are sonified within a particular reverberation volume is relatively large, they will return an echo of high intensity and the logic in the system may attribute this to a bottom echo. If so, processing of subsequent echo intensity measurements stops and the integrated echo intensity from the school of fish will be less than it should be. The fish abundance will be underestimated. If this phenomenon, *"false bottoms,"* occurs, it is usually in areas of heavy fish concentrations, so while it may be rare in terms of the total number of transmissions, its occurrence, at a time when fish echoes have their greatest effect upon the estimate of abundance can, if uncorrected, result in important underestimates of fish abundance.

The second phenomenon occurs when the bottom echo never reaches that intensity required by the system to cause its logic to declare that a "bottom echo" has been received. In this case the relatively low-intensity bottom echo is interpreted as "fish echoes", and huge numbers of relatively high intensity echo measurements can be incorrectly included in the integrated echo intensity used to estimate fish abundance. This I term *"lost bottoms."* This situation can occur in areas of steeply sloping seafloor where the seabed echo starts from the side of the reverberation volume that first sonifies the seafloor, and continues as the intersection of the reverberation volume and the seafloor moves across the face of the reverberation volume. The sea bed echo is considerably elongated but contains in total only the same energy as one from a flat seafloor (of the same acoustic impedance). The echo intensity is thus unavoidably lower than for echoes from seabeds that are more or less perpendicular to the direction of transmission.

Previous Simrad acoustic systems have permitted some operator control over these problems by allowing the operator to vary the level of the echo intensity at which a bottom echo is "declared" by the system. This was done by adjusting the Discriminator setting. Lowering the setting reduced the intensity at which a bottom echo was declared; raising it did the reverse. If watch keepers on acoustic cruises paid conscientious attention to seabed conditions, sea depth and the discriminator setting, the consequences of this problem could probably be ignored. The reality is that inevitably during cruises conditions occur at least once (usually about 0330) where the discriminator settings are inappropriate, leading to large numbers of seafloor echoes being passed to the mass storage device, or to the seafloor being interpreted as occurring near the surface of a large school.

Post cruise processing of data collected during Gulf of St. Lawrence herring (*Clupea harengus*) cruises showed that editing of the acoustic data was essential to determine if *false* or *lost bottoms* had occurred, and to allow the opportunity to make necessary corrections. Detailed examination of herring echo records shows that the reality can be even more complicated than so far described. Large herring schools are sometimes associated with discontinuities in the sea floor; thus the conditions that cause both *false* and *lost bottoms* may occur simultaneously!

It has proved impossible (to me) to contrive a single set of logic that could, without fail, distinguish when these phenomena were and were not occurring, because their effects grade into normal echo characteristics associated with seabed detection. For example, was a small jump in the seabed a *lost bottom* or a genuine abrupt change in the seafloor? The important conclusion is that no system of Bottom Window logic can be completely foolproof. Thus operational methods must be used that permit at all times a check on the reliability of the Bottom Window processing logic.

The Bottom Window logic algorithm I have used processes the fish echo records using assumptions that are parameters of the computer program to identify occurrences of *false* and *lost bottoms* in the echo record. When these phenomena occur they are corrected by insertion of echo records in the case of *false bottoms*, and by deleting records in the case *lost bottoms*. Appropriate new sea floor times are calculated and are placed into the echo record. When such adjustments occur, the program retains records of the amount of data deleted or inserted. Thus it is possible by examination of the editing results and the relevant section of the echogram to review assumptions about parameters that were used in the Bottom Window algorithm's logic. Reality is even further complicated in that occasionally, a *false bottom* will be encountered in the middle of pulses with *lost bottoms* and pulses with *lost bottoms* can be encountered in the middle of a sequence of pulses showing *false bottoms*. The program

as it exists contains some crude code to cover these situations, but the logic requires refinement.

The program I have used was written in Fortran 4 for implementation on the Cyber Series 171, located at the Bedford Institute of Oceanography. It is part of a sequence of programs used to process echo data collected by the ECOLOG System for Gulf of St. Lawrence herring stocks (Shotton 1986, Shotton et al. 1987a, Shotton et al. 1987b, Cairns et al. 1988, Cairns et al. 1989, Cairns and Wright 1990). System details are given in Shotton and Dowd (1978). A detailed description of the program logic (including the ad hoc code insertions) is given in Appendix I.

2. INPUT DATA

Digital fisheries acoustic systems measure the intensity of fish echoes and record the range of the target and the seabed. Other information may also be stored. For example, in processing echo data collected by the ECOLOG system, echo intensities from two arrays within the transducer (a wide and narrow beam) are recorded. It is possible to identify the pulse number for every pulse from the transmission reverberation splash that is detected and stored by the system. Thus each echo measurement that is processed in the Cyber consists of:

1/ raw data buffer number,

2/ echo time (microseconds),

3/ seabed echo time,

4/ A/D measure of the wide beam echo intensity,

5/ A/D measure of the narrow beam echo intensity, and

6/ pulse number.

Any sequence of echo measurements that have the same pulse number, or a consecutive pulse number, may have come from the same school. Pulses with no echoes (from herring schools) are not retained in the post cruise data record (though the raw data remain available in case a review is needed of decisions regarding what is and what is not a school). Non-consecutive pulse numbers indicate a separation between schools in the data record. The difference between any pulse numbers that are numerically non-consecutive but follow in the data record depends on the physical separation of the schools at the time of the transect and factors such as the speed of the survey vessel, pulse rate, and equivalent beam angle of the transducer.

Groups of echo measurements from individual schools are separated by file marks (EOF) in the data record which are inserted by the Cyber-based processing algorithms. This system of programs can be used for any acoustic system that stores its data in a manner that enables separate echo and pulse data to be identified. With this manner of post cruise analysis, any failure of Bottom Window logic assumptions can be identified and corrected by reprocessing the data using assumptions that are believed to be more realistic. Any failure of the Bottom Window logic where no fish data occur is of no importance as these data are not retained by earlier editing with other programs of the raw data records.

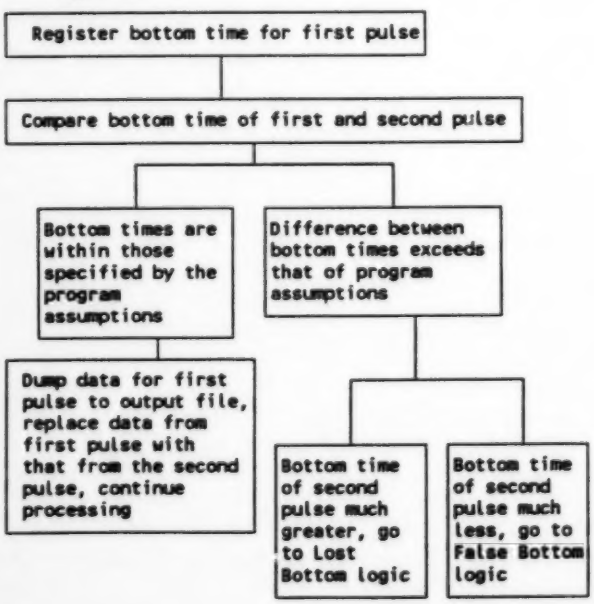
3. PROCESSING PROCEDURES

3.1 INTRODUCTION

Figure 3.1 shows an outline of the logic flow for Bottom Window processing used by the EDITAL program. A detailed flow structure is described in the program documentation in Appendix I.

A bottom time is flagged from the first pulse to sonify the school. It is assumed that this bottom is "legitimate", i.e., the school echo data do not start in a *lost* or *false bottom* condition. This bottom time is set as a reference level, to be updated by successive legitimate bottom times. The echo measurements from this pulse are read into an array. The first record of the next pulse is then read. This is detected by an increase in the pulse number. The bottom times of the successive pulses are then compared. If their difference is less than the value set as a program parameter, it is assumed that no failure of the bottom detection logic occurred in the acoustic system for that pulse. The data from the first pulse are dumped to the output file and the rest of the echo records from the next pulse are then read, overwriting the data from the previous pulse

Figure 3.1


Register bottom time for first pulse
Compare bottom time of first and second pulse
Bottom times are within those specified by the program assumptions
Difference between bottom times exceeds that of program assumptions
Dump data for first pulse to output file, replace data from first pulse with that from the second pulse, continue processing
Bottom time of second pulse much greater, go to Lost Bottom logic
Bottom time of second pulse much less, go to False Bottom logic

were stored in the same array. The seafloor time of the second pulse becomes the new reference bottom time. The bottom times of successive pulses are checked and the data handled in the same manner. If an EOF is encountered, the data for the penultimate and ultimate pulses are flushed to the output file.

If, in the comparison of the bottom times of two consecutive pulses, the bottom time of the second is less than that of the preceding pulse by amount exceeding that specified by the parameter value, a *false bottom* is declared and appropriate processing logic is followed (See Section 3.2). If the second bottom time is greater than the preceding bottom time by an amount exceeding the parameter value, a *lost bottom* is declared and control switches to the logic for that situation (See section 3.3).

3.2 *FALSE BOTTOMS*

When a pulse is encountered for which a *false bottom* is declared (i.e., an apparent upward jump in the bottom), the bottom time for the preceding pulse is stored together with the A/D values for that (*last good*) pulse. The A/D values are referenced in the storage array by their echo return times.

The data for this, the first pulse with a *false bottom* time, are written to a temporary output file, tape or logical unit 3. The next pulse is then read and its bottom time compared with the last pulse considered to have a valid bottom time. This is not done as a straight comparison; the program logic permits a possible change in the seafloor to occur equivalent to a pre-specified amount per pulse. Thus, if an absolute difference, D, exists between the "*last good* pulse" and the pulse currently being processed, and the parameter accounting for a possible slope in the seafloor has a value R, then a true bottom is declared to have occurred when

$$P_c - P_{lgp} > D / R \qquad (3.1)$$

where P_c = pulse number for the data currently being processed,

P_{lgp} = pulse number of the "*last good* pulse",

D = absolute difference in bottom times between the two pulses, and

R = "Force Return" or sloping-bottom program parameter.

To recapitulate, echo data are read from the input file (unit 1) and, if declared acceptable, are written to the output file, unit 2. A/D values, referenced by their echo times, are stored in an array for later use. The records for pulses deemed to have *false bottoms* are stored in a temporary file - unit 3. When records from a pulse whose bottom is deemed to be "good", (i.e., it satisfies Equation 3.1), are reached, two tasks are done:

(1) estimate the bottom time for each pulse where a *false bottom* was deemed to have occurred, and,

(2) interpolate echo data into the echo record for each pulse, from the time at which the *false bottom* occurred, down to the true bottom as calculated.

An estimated "true" seafloor time is obtained by simple linear interpolation using data from the *"last good* pulse" prior to the *lost bottom* and the *"first good subsequent* pulse" after the sequence of lost bottom pulses. Thus

$$BT_{LG+i} = BT_{LG} + i\left(\frac{BT_{LG} - BT_{FSG}}{P_{FSG} - P_{LG}}\right) \qquad (3.2)$$

where:

BT_{LG+i} = calculated bottom time of the i^{th} pulse after the *last good* bottom time,

BT_{LG} = bottom time of the *last good* pulse before the *false bottoms* sequence,

BT_{FS} = bottom time of the *first subsequent good* pulse after the *false bottoms* sequence,

P_{FSG} = pulse number of the *first subsequent good* pulse,

P_{LG} = pulse number of *last good* pulse.

For example, if bottom times are as follows:

BT_{LG} = 100,

BT_{FS} = 120,

P_{LG} = 10,

P_{FSG} = 15,

$$\text{Then } BT_{LG+i} = 100 + i\left(\frac{120 - 100}{15 - 10}\right)$$

and

BT_{LG+1} = 104 ... etc.

The data from pulses with *false bottoms* are now read from the temporary file (unit 3). The *false bottom* time in each echo measurement record is replaced with a new time as given by 3.2.

When the last record for each pulse is reached, an interpolation procedure calculates A/D values for records that are added to the pulse record sequence from the (old) *false bottom* time to the new bottom times that have been calculated using (3.2). These interpolated A/D values are derived from the A/D values in the *last good* pulse and the *first subsequent good* pulse at the corresponding time. For each successive time in this sequence, a new record is inserted if an A/D value existed in the *last good* pulse and/or the *first subsequent good* pulse at the same time (i.e. range). Thus for times such that,

$t_e > t_{FB,i}$,

$$V_t = \frac{V^2_{LGt} + V^2_{FSGt}}{2} \qquad (3.3)$$

Where,

t_e = time echo received,

$t_{FP,i}$ = time of *false bottom* in i^{th} measurement,

V_t = voltage value inserted at time t,

V^2_{LGt} = voltage value at time t in *last good* pulse,

V^2_{FSGt} = voltage value at time t in the *first subscquent good* pulse.

In an earlier version of this program, an arithmetic mean of the A/D values was used. This would overestimate the mean (interpolated) echo intensity. Note that should there be no corresponding A/D value at time t in the "*last good* pulse" or "*first subsequent good* pulse, (3.3) should still be used but with V^2_{LGt} or V^2_{FSGt} set to zero as is appropriate. In such cases in the program the A/D value was divided by two; this estimate is biased and is an underestimate of the mean echo intensity value.

If no values occurred at time t in the *last good* pulse and in the *first subsequent good* pulse, no additional records are inserted into the echo sequence in any of the pulses so processed. This is not a particularly elegant interpolation procedure, but in reality, a much more complicated procedure would probably not result in a very different abundance estimate.

Should an end-of-file (EOF) be encountered in the data stream before a pulse is encountered that is deemed to have a good bottom time, the interpolation routine is based solely on the bottom time and A/D values contained in the *last good* pulse. Thus the A/D values inserted into each pulse beneath their *false bottom* times will be identical to the A/D values in the *last good* pulse at the corresponding echo time. An improved interpolation procedure might consider more than solely the two adjacent (before and after) good pulses.

3.3 *LOST BOTTOMS*

A similar logic is used in this situation to that described in 3.2 but the processing is simplified because no data are inserted; data, which this logic procedure deems to be from the sea floor, are only dropped from the echo record. When a pulse with *lost bottom* is encountered, the bottom time of the *last good* pulse is stored and the echo data are read from the input file until a pulse that satisfies 3.1 is encountered. In records deemed to have a "*lost bottom*", seafloor times are replaced with a new seafloor time calculated by Equation 3.2. The data for each pulse are then written to the output file (unit 2) on the condition that for each record, the echo time is less than the newly calculated bottom time. Records with an echo time greater than the new bottom time are ignored (the logic loops past them). A procedure can be used at this point to reject echoes that are deemed to be occurring in the bottom rise time.

4. MONITORING OF PROCESSING

As mentioned in the Introduction, experience with the editing for *lost* and *false bottoms* shows that invariably assumptions regarding what are appropriate parameters values will in some instances be incorrect, i.e., a *false bottom* will remain undetected, or a *lost bottom* will be wrongly declared. The best guard against this is to monitor the consequences of the Bottom Window processing logic and to check it against the echogram to ensure that the assumptions have been working, at least at an acceptable level. To facilitate this, for each school, the following data are printed to a summary file that can be inspected after an editing run:

1) number of records read (from unit 1),
2) number of records written (to unit 2),
3) number of records inserted,
4) number of records dropped,
5) number of A/D measurements in which every A/D bit was set - an indication of system saturation.

6) first pulse number for the school,

7) last pulse number for the school,

8) records attributed to noise (a continuous sampling sequence within a single pulse starting soon after transmission, an occasional idiosyncrasy of the ECOLOG system).

9) the number of *lost bottom* records dropped,

10) the percentage of records in which the narrow beam echo intensity exceeded the wide beam echo intensity.

In addition, for any run, the program processing parameters (e.g., those determining when a *lost bottom* or *false bottom* occurred) are also recorded to identify the values used in a particular edit run. These are:

1) date of the edit, from the Cyber calendar,

2) the number of time units from the bottom within which records were deemed to be part of the bottom rise. Generally a value of 3 or 4 was used (1 unit = 7.5 cm).

3) Number of time units by which successive pulse bottom times must differ to trigger *false* or *lost bottom* processing. Usually a value of 18 was used, i.e., 1.35 m between successive pulses.

4) *False bottom* "Force-return" parameter. Generally a value of 5 (37.5 cm per pulse) was used,

5) *Lost bottom* "Force-Return" parameter. Generally a value of 6 was used (45 cm per pulse).

5. DISCUSSION

My general experience is that the bias in total estimates that would be caused by *false* or *lost bottoms* during a cruise is not large, but when it does occur, e.g., in a particular stratum, it can cause considerable error, particularly if discriminator settings have not been carefully monitored (the black box syndrome!).

I strongly endorse the practice and concept of off-line processing for correcting Bottom Window logic failures. Only in this way can the effects of changes in the assumptions of parameter values be determined. The amount of computer processing needed to do this has not proved to be excessive. Off-line processing also requires that the data be retained as originally collected. Such considerations underline the importance of raw data storage in future acoustic systems.

The EDITAL Program was written under operational pressure, i.e., urgency to get some valid results. It is overdue for scrutiny and rewriting, possibly for implementation on a PC (in Turbo Pascal, for instance). The code layout can be improved, some errors corrected, and null code and code used to force logic jumps in some unusual conditions could be removed. Consideration should also be given to improved bottom time and A/D interpolation methods. Despite this, the program seems to have served its function.

6. LITERATURE CITED

Cairns, D.K., and J.A. Wright. 1990. Distribution and acoustic backscatter of herring in NAFO Division 4T, November 1989. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 90/72.

Cairns, D.K., J.A. Wright, and E.M.P. Chadwick. 1988. Acoustic backscatter estimates and school behaviour of herring in NAFO Divisions 4T and 4VN, November 1987. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 88/46.

Cairns, D.K., J.A. Wright, and E.M.P. Chadwick. 1989. Distribution and acoustic backscatter of herring in NAFO Divisions 4T and 4VN, November-December 1988. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 89/42.

Shotton, R. 1986. Results of the acoustic survey of herring in the southern Gulf of St. Lawrence and Sydney Bight, November 1985. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 86/84.

Shotton, R., M. Ahrens, and C. Bourque. 1987a. Results of an acoustic survey of herring in the southern Gulf of St. Lawrence and Sydney Bight, November 1984. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 87/63.

Shotton, R., M. Ahrens, and C. Bourque. 1987b. Results of the acoustic survey of herring stocks in NAFO Divisions 4T and 4Vn November 1986. Can. Atl. Fish. Sci. Adv. Comm. Res. Doc. 87/63.

Shotton, R. and R. G. Dowd. 1978. The development of a system for acoustic assessment of fish stock size. Proceedings of the Conference on Oceanographic Data Systems, Woods Hole Oceanographic Institution, Woods Hole, Massachusetts, USA.

Appendix I
Program Variables

IP Loop variable that counts the number of echo measurements within a pulse.

IPULSE(1600,6) Array that holds the data from each successive pulse. It is dumped to unit 2 if the data are deemed to have a good bottom time, if not it is dumped to the temporary holding file, unit 3. Up to 1600 records per pulse can be stored. This can be changed depending on memory considerations.

LINE(6) Each record from the input file is read into this array. If the record comes from the same pulse as the previous pulse it is stored in IPULSE.

INLINE Count of the total number of lines read from the input file.

INSERTS Count of the number of records inserted into the output sequence during *false bottom* processing.

IRJCTS Number of records dropped as being *lost bottom* records.

NUMPUL Number of pulse sonifying the school.

NOVER Number of A/D measurements with all bits set, i.e., with value equal to 4095.

NBOTTS Count of the number of records that are rejected because they are deemed to be in the bottom rise time.

IREJ Number of time units attributed to bottom rise time, i.e., only records with echo times < (Bottom Time - IREJ) are accepted.

ISTEP Number of time units by which a consecutive bottom time must differ to start the bottom correction processing.

IFALRET Average number of time units per pulse by which successive pulse times may differ for the code to declare a return to a *good bottom* time.

LOSRET As for IFALRET above, but for *lost bottoms*.

NRUN School or file count.

IEOF End of file flag. Value depends on unit monitored.

NSERR Count of noise records within a school.

LUNIT File variable, either the output file (2) or the temporary file (3).

ISTEP When *lost bottom* processing is triggered, this stores the time difference of the bottoms of the *last good* pulse and the *first subsequent good* pulse.

IOLDPUL Stores "*last good* pulse" number.

IOLDBOT Stores "*last good bottom*" time.

ICNTRL An end of file flag in the subroutine FALSEBOT. Identifies unit of the status check.

NSERTS A temporary count of inserted records when a *false bottom* is detected within a *lost bottom*.

BOTINC Increment on bottom times when inserting new bottom times in echo records.

IW1 Temporary array storing A/D values from the wide angle transducer array from the *last good* pulse.

IW2 As above but for the *first subsequent good* pulse.

IN1 As for IW1, but for the narrow beam transducer elements.

IN2 As for IN1 but for the *first subsequent good* pulse.

LASTB Stores the calculated bottom time to be written to specified rows in IPULSE[i,3].

Appendix II
Program Logic Description

The version I have in Bissau where this was written is in FTN4.

AII.1 Main Program

1. Set IREJ variable for rejecting bottom rise time echoes.

2. Set ISTEP variable that flags minimum time difference required to start Bottom Window correction processing.

3. Set IFALRET, bottom slope pulse multiplier that

forces return from *false bottom* assuming a sloping bottom.

4. Set LOSRET which does the same as IFALRET but is used when a *lost bottom* has occurred.

5. Set End-Of-File flag and school count to zero (n.b. multiple assignments are not permitted in FTN5).

6. Print date to default output using CDC Fortran routine.

7. Call HEADER (printing) subroutine.

8. Increment school counter (NRUN = NRUN + 1).

9. Print run (school) number to default output file.

10. Set following to zero for the current school:
    number of lines inserted,
    number of pulses in school,
    number of A/D measurements with all bits set,
    number of records written to output file,
    number of records rejected in *lost bottom* routine,
    number of records rejected as noise,
    number of records where $V_w < V_n$,
    number of records rejected as being in bottom rise time.

11. Set lines read to 1.

12. Read first line of school data (consisting of buffer number, echo time, seafloor time, wide beam A/D, narrow beam A/D, and pulse number) into IPULSE[1,j],j=1,6.

13. Stop if the end of all data is encountered, i.e., a double EOF or End-Of-Tape mark.

14. Store pulse number in IPUL.

15. Call Subroutine READLN to read in remaining records from that pulse into IPULSE.

16. Check if EOF flag was set while in READLN; if yes go to DUMPDAT.

17. Check that the next pulse is consecutive. This should be redundant, depending on the processing of the echo data prior to running EDITAL, i.e., it should always return true. If not, all records in IPULSE are dumped and a new school is started. A jump of more than one in the pulse number would indicate that a subsequent school had been encountered.

18. The bottom time is compared with that for the next pulse (LINE[3]). If ISTEP < = modulus (Last Good Bottom Time - bottom time), the data in IPULSE are dumped to the output file and the remaining records in the next pulse are read into IPULSE.

19. If the absolute difference is > ISTEP, the direction is checked to determine if a *false bottom* or a *lost bottom* has occurred.

20. If a *lost bottom* has occurred, the data are checked to see if it is noise (this is an empirical check). When these noise echoes occur the first record is typically within 30 time units of the transmission and the difference in the bottom time units is >700, again an empirical comparison. If the check returns False, i.e., no code skip occurs, the noise counter is suitably incremented by the number of records in IPULSE i.e., IP, these data are skipped.

21. The first line in IPULSE is set to LINE; the first record of the subsequent pulse read in.

22. In the case of a *lost bottom*, processing switches to the LOSTBOT subroutine. The parameters are the EOF flag, LOSRET and the counter of the number of records rejected.

23. On return to the main program, the EOF flag is checked. If True, processing is transferred to the end-of-school summary processing in Subroutine HEADER; if False, Subroutine READLN is called again to read in the data for the next pulse.

24. If a *false bottom* has been detected, processing is switched to the Subroutine FALSBOT.

25. On return, the EOF flag is checked; if it has the value False, processing loops back to check for another erroneous bottom in the following pulse. If True, processing switches to the end-of-school processing in subroutine HEADER.

26. The EOF flag is turned off.

27. Processing returns to the beginning of the code.

A2.2 Subroutine READLN

This subroutine reads in all the data from a single pulse into IPULSE. The input file can be the raw data file (unit 1) or the temporary storage file (unit 3). Unit 3 is used when a bottom error condition is detected and a temporary dump is needed to store data while input continues until it is deemed that a good bottom time is once again encountered.

1. The record counter is set to 1. The first record has been read into LINE in the main program.

2. The next record is read.

3. The file status is checked, if False,

4. the INLINE records read counter is incremented.

5. If the record is from the next pulse, control returns to the main program.

6. If not, the record counter for the pulse, IP, is incremented and,

7. the contents of the LINE are transferred to the next (vacant) row of IPULSE.

8. Control then loops back to read the next record.

9. If an EOF is encountered or the raw data file, the flag is set to 1, and to 3 if the EOF was encountered in the temporary storage file, unit 3.

A2.3 Subroutine DUMPDAT

This subroutine writes the contents of IPULSE, i.e., the data for a single pulse, to either the output file (unit 1) or the temporary file, unit 3. The appropriate counters are incremented.

1. The logical unit flag is set to zero.

2. A loop is entered that processes each record in IPULSE, starting with the first record measured in the pulse, i.e., the target closest to the transducer.

3. If the record is within the bottom rise time, and normal bottom processing is occurring, i.e., LUNIT = 2, control switches out of the loop.

4. Otherwise the record is written to the temporary storage file, unit 3.

5. If LUNIT = 2, i.e., the edited data output file, NOUT is incremented for each record written.

6. If all A/D bits are set, NOVER is incremented.

7. If $V_n > V_w$, NBIG is incremented, i.e., the intensity received on the narrow beam is greater than that received on the wide beam.

8. LOUT is set to 1.

9. If echoes were recorded in the bottom rise time, RJCTS is incremented by the number of records in this period.

10. If LOUT was set in the loop and LOUT = 2, i.e., the data are good, the pulse count variable NPUL is incremented.

11. The first record of the next pulse is stored in the first row of IPULSE and,

12. the record count indicator, IP, is set to 1.

13. Control is returned to the main program.

A2.3 Subroutine LOSTBOT

1. A message is flagged to the output file to indicate that this subroutine has been called and the first record and the bottom time difference also written.

2. This difference is stored in ISTEP.

3. The pulse number of the *last good* pulse is stored in IOLDPUL.

4. The *last good* pulse bottom time is stored in IOLDBOT.

5. The data with good bottom times in IPULSE are dumped to the output file (unit 2) through a call to DUMPDAT.

6. The rest of the data for the *first bad bottom* pulse are read into IPULSE.

7. An EOF check is done. If True, control switches to subsequent code (statement 500).

8. An empirical logic check here has been used to force returns when the bottom time difference that has been encountered is less than one third of the initial step. This line of code should be dropped in future version of the program.

9. The number of pulses from the *last good* pulse to the pulse currently being processed is multiplied by the force-return value in LOSRET, i.e., the variable accounting for the possibility of a sloping bottom. If this value is greater than the current bottom time it is deemed that the current bottom is "good" and code is switched ahead to handle this situation.

10. Otherwise, the data are written to the temporary file, unit 3 and the data for the next pulse read in.

11. When data are encountered in the input file that are deemed to have a *good bottom* time, a message is written to the default output file, and,

12. new bottom times are calculated for the intervening pulses by calculating a bottom time increment to be added to the *last good bottom time.*

13. The last of the records with erroneous bottom times are dumped to unit 3 which is then rewound.

14. The *first subsequent good* record is stored in LINE, and,

15. the first record in unit 3, the temporary file, is read into (IPULSE[1,j],j=1,6).

16. IP is set to 1.

17. The rest of the data for the "*first bad bottom*" pulse are read from unit 3.

18. A loop is executed that substitutes a new bottom time in IPULSE[I,3], calculated from the "*last good bottom*" time and the bottom increment (BOTINC) that has been calculated. A round off factor of 4.51 is also added to account for integer round off and rejection in the bottom rise time, i.e., the inserted records are entered above a later bottom-rise time check. This needs further consideration.

19. If the echo time exceeds the calculated bottom time excluding the 4.51 round-off factor, control switches out of the loop.

20. After the loop, the number of records rejected is added to the counter IRJCTS.

21. IP is set to one less than the count value on leaving the loop as the test is done after I has been incremented in the loop, and,

22. the data are dumped to the output file, unit 2.

23. An EOF check is done to determine whether to loop back and continue processing.

24. If EOF = 3, an EOF has been encountered on the temporary file, unit 3; the EOF flag is then reset. [This line should be dropped - it is null code].

25. The *first subsequent good* Line can now be transferred to (IPULSE[1,j],j=1,6) and control returns to the main program. [This line should be dropped - it is null code].

26. When an EOF is encountered in the raw data input file before a *first subsequent good* pulse is found, the following code is executed:

27. Unit 3 is rewound,

28. The EOF flag is reset,

29. The first line from unit 3 is transferred to IPULSE[1,j].

30. A call to READLN retrieves the rest of the data for that pulse into IPULSE.

31. Control cycles through these records changing IPULSE[I,3] to IOLDBOT, i.e., the *last good bottom* time. If the echo time in IPULSE(I,3) exceeds IOLDBOT, control switches out of this loop and the count of the records output is incremented.

32. The number of records rejected as *lost bottomers* is counted, and IP, the number of records to be written to unit 2, is set to the loop counter, I.

33. The data are dumped to unit 2.

34. An EOF check is done. When an EOF is encountered, IOF is set to 3 in the subroutine READLN and is then reset to 1 in the subroutine LOSTBOT. This triggers an unconditional transfer with control returning to the main program.

A2.4 Subroutine FALSBOT

1. The records inserted (NSERT) are counted and a logic flow control flag (ICNTRL) is set to zero.

2. A message is written to the output file to indicate that a *false bottom* within the school is occurring.

3. The *last good* pulse number and the *last good bottom* time are stored in IOLDPUL and IOLDBOT respectively.

4. An array bound dimension check is done, and,

5. The temporary array that stores the A/D_w and A/D_n values for the "*last good*" pulse and "*first subsequent good*" pulse are set to zero.

6. The *last good* pulse A/D values are stored in the IW1 and IN1 arrays. The row number is directly indexed by the echo time.

7. The temporary file, unit 3, is rewound and the data in the *last good* pulse (stored in IPULSE) are dumped to the output file, unit 2.

8. The rest of the pulse data are read into IPULSE.

9. An EOF check is done. If true, the school data finished before a pulse with a *good bottom* was encountered and processing jumps to line 480 to handle this situation. If the EOF is "False" those data are dumped to unit 3.

10. The bottom time in the next pulse is checked to see if a *good bottom* has occurred. If not, control loops back, the records for the pulse are dumped to unit 3 and another input cycle occurs.

11. A subsequent empirical check is done. If this check determines that the new bottom time is more than 200 time units smaller than the *last good bottom* time it is deemed that a *false bottom* has occurred in the middle of a *lost bottom*. A flag is set (ICNTRL = 2) and processing is switched to line 480. This code need refining.

12. Otherwise, it is considered that a *good bottom* has been encountered.

13. The remaining records for that pulse are then read in.

14. A check for an EOF is done. If true, ICNTRL is set to 1.

15. The change in bottom depth per pulse from the *last good* pulse to the *first subsequent good* pulse is calculated.

16. Unit 3, the temporary storage file is rewound and the number of records that were in the *first subsequent good* pulse is stored in NP.

17. The data comprising the *first subsequent good* pulse are stored in the array JPULSE which has an identical structure to IPULSE. This is necessary as IPULSE is used as the temporary store when data are retrieved from unit 3 for processing.

18. As part of the same loop, the A/D values, indexed by their corresponding echo times, are stored in IW2 and IN2, i.e., arrays that match IW1 and IN1, but provide a record of the A/D values in the *first subsequent good* pulse.

19. Likewise, the array LINE stores the first record of the pulse that follows the "*first subsequent good*" pulse and is transferred to NLINE for future reference.

20. The first line in unit 3 is read into (IPULSE[1,J]J = 1,6), then the subroutine READLN is used to read in the rest of the records for that pulse.

21. The (*false*) bottom time of these records is stored for later use as a loop control variable and a new bottom time (LASTB) is calculated, taking into account BOTINC, the change in the bottom slope.

22. The seafloor times in IPULSE (IPULSE[i,3]) are changed to the new bottom time.

23. The next section may insert new records into that particular pulse's data, starting with times that are greater than the old (*false*) bottom time and continuing to the new bottom time that has been calculated. Whether a record is inserted at a particular range depends on the existence of a record at the same range in either the *last good* pulse or the *first subsequent good* pulse, i.e., the pulses that precede and follow the sequence of pulses with *false bottom* times. If an A/D measurement was taken at a particular time(s), an average value is calculated using (3.3) and a new

record is created and appended to the pulse record sequence. The count of the number of records inserted is incremented. Should an A/D value at a particular range exist only in either the *last good* pulse or the *first subsequent good* pulse, then the inserted record will have A/D values equal to half those of the comparison record. The number of records is now greater than the pulse record count, IP, so it is updated to include the added records (IP = J).

24. The data for the expanded pulse are then dumped to the output file.

25. If the EOF flag is set this indicates that the data in unit 3 have all been processed. The data for the *first subsequent good* pulse are returned to IPULSE from JPULSE and,

26. the first line of the pulse after the "*first subsequent good*" pulse is retrieved from NLINE and returned to LINE. The pulse record count, stored in NP, is transferred to IP.

27. If EOF = 3, i.e., all data were read in from unit 3, IEOF is set to 0, but if ICNTRL has been set indicating that an EOF has been encountered on the raw data input file, IOF is set to 1 to prevent further attempts at reading the (exhausted) input file.

28. Finally, the count of lines that are inserted is incremented and control returns to the main program.

The remaining code in this subroutine handles situations where an EOF is encountered when a *false bottom* condition exists, or a *lost bottom* has occurred in a *false bottom*.

29. The first record of the last pulse (LINE) is transferred to a temporary array, NLINE, and the temporary output file rewound.

30. The first record stored in unit 3 is read into IPULSE[1,j], the pulse record counter, IP, is set to 1, and the EOF flag cleared.

31. The remaining records for that pulse are read in through a call to READLN. This code is not consistent with earlier practice.

32. A bottom variable is set to a value one greater than the *false bottom* seafloor time. This becomes a loop control variable for data correction.

33. The records in IPULSE have their bottom times reset to IOLDBOT, the "*last good bottom*". The number of records in the pulse is stored in J, then a loop is executed starting at K and finishing at IOLDBOT.

34. If there was a record at any of the bottom times between (and including) K and IOLDBOT, as indicated by a non-zero A/D value in the array of A/D_w values stored from the "*last good*" pulse, a new record is created with identical A/D_w and A/D_n values to the records at the corresponding range in the *last good* pulse, in the range K to IOLDBOT.

35. The count of lines inserted is incremented and the data are dumped to the output file.

36. If an EOF was encountered on unit 3, i.e., its records were all read in, and ICNTRL = 2, indicating that a *false bottom* had occurred in a *lost bottom*, program flow jumps forward. The count of lines inserted is incremented and the first record of the "*first good*" pulse data are restored to IPULSE[1,j]; control is returned to the main program.

37. If the EOF flag is set to 0, control goes to the subroutine READLN and the data for the next pulse are read from unit 3.

38. The last EOF condition, IOF = 1, returns control to the main program for End-of-File processing after the line counters have been incremented.

8. Subroutine HEADER

This subroutine writes the data pertaining to the relevant bottom processing of each school to the default output file. It is first called before the school counter has been incremented. In this call when NRUN = 0, the processing parameters are listed together with a data header. At this time the following are written to the default output file:

Date,

Default number of time units adjacent to the bottom for which data will be rejected,

Number of time units used as test for a *lost bottom* (ISTEP),

Multiplier per pulse (to account for bottom slope) to force a return from the Bottom Window correction

code when a *false bottom* is encountered,

The same as the above but for a *lost bottom* condition.

A header creates column captions for:
run (school) number,
lines read in,
lines written,
lines inserted,
lines deleted,
number of pulses in school,
number of records with all A/D bits set,
first pulse number,
last pulse number,
number of records attributed to noise, and,
number of records with $V_n > V_n$, written as a percentage.

Finally an EOF is written to the output file to delimit the individual school's data.